Homework IV

Due date: Wednesday, Apr 3rd 2024 at 11:59pm

Note: For written questions, you can turn in either a scanned copy of your handwritten answers or a PDF file of your answers. For programming questions, you need to submit your code and **also answer the asked questions in pdf**. Name the submission package as hw4_Lastname_Firstname.zip.

Problem 1: Naïve Bayes for Text Classificaiton (15 points)

Consider a Naïve Bayes classifier modeling **word count** for a binary classification (class y=-1 and class y=1) problem. There are three words of interests, namely w_1, w_2, w_3. The probability of observing a word in each class is given in the following table. For example, this tables reads

Table 1: Conditional probability of observing each word in documents of a class.

	w_1	w_2	w_3
$y = -1$	1/10	2/10	7/10
$y = 1$	5/10	2/10	3/10

as $Pr(w_1|y = -1) = 1/10$. That means in documents from $y = -1$ class, the probability to observe word w_1 is 1/10. The prior probabilities for the two classes are $Pr(y = -1) = 4/10$, and $Pr(y = 1) = 6/10$. Answer the following questions.

1) Consider the documents with word counts $x = (1, 0, 1)$, i.e., count for w_1 is 1, count for w_2 is 0, count for w_3 is 1. Which class has the highest posterior probability?

2) Consider the documents with word counts $x = (2, 0, 1)$, i.e., count for w_1 is 2, count for w_2 is 0, count for w_3 is 1. Which class has the highest posterior probability?

Problem 2: Implementing Naïve Bayes for Text Classification (50 points)

Method In this problem, you are asked to implement a Naïve Bayes Classifier for text topic classification. **No machine learning library is allowed in this question, yet you can use basic libraries such as Numpy for data structures.** Suppose there are m distinct words ($\{w_1, w_2, ..., w_m\}$) in total within the training data. We represent a document $\mathbf{x}$ by $\mathbf{x} = (x_1, x_2, \ldots, x_m)$, where x_j is the occurrence frequency of word w_j in this document $\mathbf{x}$.

Given a collection of training documents $\mathbf{x}^1, \ldots, \mathbf{x}^{n_k}$ in the class C_k, where $\mathbf{x}^i = (x_1^i, \ldots, x_m^i)$, the probabilities of $p(w_j|C_k)$ can be estimated by MLE, i.e.,

$$p(w_j|C_k) = \frac{\sum_{i=1}^{n_k} x_j^i}{\sum_{j'=1}^{m} \sum_{i=1}^{n_k} x_{j'}^i}, \forall j, k. \tag{1}$$

To avoid the issue that some words may not appear in training documents of a certain class, the estimated probabilities are usually smoothed. One smoothing method is Laplacian smoothing which computes $p(w_j|C_k)$ by

$$p(w_j|C_k) = \frac{\sum_{i=1}^{n_k} x_j^i + 1}{\sum_{j'=1}^{m} \sum_{i=1}^{n_k} x_{j'}^i + m}, \forall j, k. \tag{2}$$

Then, for a class C_k and a data point $\mathbf{x}$, we model $\Pr(\mathbf{x}|C_k)$ as

$$\Pr(\mathbf{x}|C_k) \propto \prod_{j=1}^{m} [p(w_j|C_k)]^{x_j}$$

where $p(w_j|C_k)$ stands for the probability of observing the word w_j in any document from the class C_k.

The log-likelihood for a data point $(\mathbf{x}, y = k)$ is given by

$$\log \Pr(\mathbf{x}, y = k) = \log \frac{\Pr(\mathbf{x}|C_k)\Pr(C_k)}{Pr(\mathbf{x})} = \underbrace{\sum_{j=1}^{m} x_j \log p(w_j|C_k) + \log \Pr(C_k)}_{f_k(\mathbf{x})} + const$$

where $\Pr(C_k)$ can be estimated by $\Pr(C_k) = \frac{n_k}{\sum_{k=1}^{K} n_k}$ and $f_k(\mathbf{x})$ can be considered as a prediction score of $\mathbf{x}$ for the k-th class. The class label of a test document $\mathbf{x}$ can be predicted by $k^* = \arg\max_{1 \le k \le K} f_k(\mathbf{x})$.

Dataset The data set for training and evaluation is the 20NewsGroup data, which is included in the provided zip file. You will find six **text** files in this data set: train.data, train.label, train.map, test.data, test.label, and test.map, where the first three files are for training data and the last three files are for testing data. In the train.data file, you will find the word histograms of all documents; each row is a tuple of format (document-id, word-id, word-occurrence-frequency). The class labels of training documents can be found in train.label with the order corresponding to training documents' id, and the topic of each class can be found in train.map. Similarly, the word histograms and the class assignments of test documents can be found in test.data and test.label, respectively. A skeleton code is given in "p2.py".

Requirements 1) The current Naïve Bayes classifier was built up without using Laplacian smoothing, i.e., it currently uses equation (1). You need to locate the right place in the code to incorporate Laplacian smoothing as equation (2).

2) Apply the learned classifier to predict the class labels for the test documents. Report the classification accuracy over the test documents (i.e., the proportion of test documents that are classified correctly).

Problem 3: Regularized Logistic Regression (35 points)

Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the training examples, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{-1, +1\}$. The negative log-likelihood of the regularized logistic regression, denoted by $L(\mathbf{w})$ is written as

$$L(\mathbf{w}) = C \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{x}_i^\top \mathbf{w})) + \frac{1}{2}\|\mathbf{w}\|_2^2$$

where C is a parameter that controls the balance between the loss and the regularization. The optimal solution for $\mathbf{w} \in \mathbb{R}^d$ is obtained by minimizing $L(\mathbf{w})$.

- Train and test a regularized logistic regression model on the sonar data set which is available here [1]. We will use the scaled version for our experiment. A copy of the data is also enclosed in this homework. Use the provided training/testing splitting. In particular, the file "sonar-scale-test-indices.txt" contains the indices of examples in the original file for training, and "sonar-scale-test-indices.txt" contains the indices of examples in the original file for testing. You can use the logistic regression of sklearn library[2]. A skeleton code is given in "p3.py". You are required to:
  - Use the 5-fold cross validation method to decide the best value of the parameter C. The candidate values for C are $0.01, 0.1, 1, 10, 100, 1000$. For each C, report the training error and validation error. Choose the best C that yields the lowest validation error.
  - Use the selected best C value to train a logistic regression model on the whole training data and evaluate and report its performance (by accuracy) on the testing data.
  - Report the results.

[1] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#sonar`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`